Introduction

This is a presentation on some thoughts and idea I had – how to implement a filter without coefficients or delay elements, and envelope detector with if statements.

# Lo-Fi DSP: Low-pass filter (sort of)

## The filter

I modeled the filter like an analog low-pass filter, since it is also in continuous domain, even though it is analog. I am well aware that this is not how capacitors work exactly in reality, however, I am presenting a prototype and further improvements may be done in the future in order to obtain better results. To obtain the model I thought in abstract – what happens to the voltage, when passing though the filter?

So if we have an analog low-pass filter and we apply a step input, then the capacitor will start charging according to its time constant, and after some time it will reach the step level. Similar thing happens if the capacitor has initial charge and then is connected to a circuit – the capacitor discharges according to its time constant until it reaches the lowest voltage possible. Having in mind the above mentioned although simple description of a capacitor, I created the following model:

for m=1:N

if x(m)>cap

cap=cap+p;

elseif x(m)<cap

cap=cap-p;

elseif x(m)==cap

cap=cap;

end

yr(m)=cap;

end

Code Lo-Fi filter Matlab code

The ‘cap’ is the initial charge of the capacitor, p is the step at which the capacitor will change its ‘charge’, ‘x’ is the input and ‘yr’ the output. Now let’s put ‘p’ to 0.02 and apply a square wave with frequency of 5 Hz (the frequency[[1]](#footnote-1) of the input is just an example.)



Figure Filter input compared to its output

From figure 1 it is clearly seen that the output contains slopes connecting its highest state with its lowest. The common sense of an engineer will suggest that certain harmonics are being attenuated in a similar fashion as a low-pass filter would. This is confirmed by comparing the frequency domain of the input and the output.



Figure Spectral comparison between the input and the output of the filter

So far so good, but how does ‘p’ exactly affect the outcome. The answer to this question is revealed in the next figure.



Figure Comparison between the input and outputs filtered through the filter with step constants of 0.1, 0.01 and 0.001

It can be observed that the lower the ‘p’ goes, the more attenuation there is, however, notice that when p=0.001 the output waveform is still triangle, but it is attenuated. This indicates that unlike traditional filters, which would ideally remove all harmonics from the input square wave, and leave only the fundamental sinewave (considering that this is the effect we would like to obtain), this one leaves the fundamental and odd harmonics with certain magnitude related to the magnitude of the fundamental in order to form a triangle waveform [1]. This also means that even if pure sinewave is applied to the input, the output will be a triangle wave.

Let’s see how does the filter reacts to a noise riding on top of a sine wave with amplitude of 1



Figure Filtered noisy sine wave, p=0.01

Figure 4 proves that the filter can be applied in noise removing applications. Moreover is can be used to find the mean of a stochastic signal. Hera is a comparison between the model and an exponential average filter for equal values of ‘p’ and ‘alpha’.

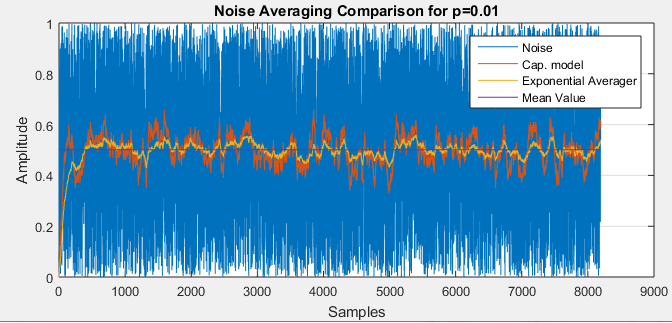


Figure Comparison between the capacitor model and the exponential average for value of 0.0

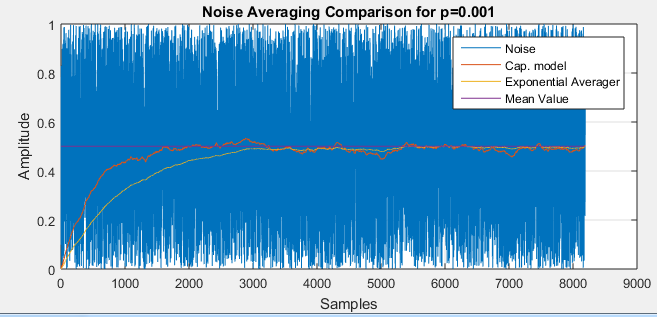


Figure Comparison between the capacitor model and the exponential average for value of 0.001

It is quite noticeable that the exponential average filter has a better response, however, have in mind that the proposed solution does not use any multipliers or memory elements.

## Parameters and variations

At this point only two variations exist, which give almost the same result: Raw Mode and Soft Mode. So what is the difference?

Whenever the input settles of a certain level, let’s say it is 1, then the output will oscillate around that value with the frequency of Fs/2 and amplitude of ‘p’. Let’s take a detailed look of figure 1.



Figure a closer look of figure 1

Notice the oscillations that occur when the output filter reaches the value of the input. This is avoided by modifying the code shown before. The new one looks like this:

for m=1:N

if x(m)>(cap+p/2)

cap=cap+p;

elseif x(m)<(cap-p/2)

cap=cap-p;

else

cap=cap;

end

ys(m)=cap;

end

Code Modified filter Matlab code (Soft Mode)

As seen from Figure 5, the modified version (Soft Mode) does not have the parasitic oscillation.

No matter which variant is used, a triangular output waveform can be obtained at best. Improvements of the routine will take place in the future.

At this point you may be asking how to choose a value for ‘p’. To answer this one more variable must be introduced- ‘amp’, which defines the desired amplitude of the triangle wave (the output) at Fc, considering that Fc is known parameter.

So let’s choose ‘p’ for Fc=5, amp=1 (peak) and Fs=2048.

Now we would like to have peak of 1, which means that for half period, the output should be able to reach from –peak to +peak. Therefore to find the step size:

Or

Have in mind that the formula for ‘p’ is just an approximation.

# Envelope and Peak detection

The second procedure is an envelope detector. Once again it is made out of if statements, however, to make it work I also had to add a 3 tap delay. The idea is simple: use the delay line of 3 elements to detect whether or not the input signal is decreasing and whenever a certain configuration of samples is detected, the highest value in the delay line is used to create the envelope of the signal. The idea may seem a bit obscure at this point, but to make it clear I give some examples below:

Let’s take as input a stream of numbers: 0, 0, 0, 1, 0, 1, 2, 2, 2, 1, 0, 1, 0, 0. Next we start to analyze the stream by taking 3 sequential numbers and comparing them to each other, therefore we star with the three 0s, since they are the first three numbers in out input.

Table Input sequence analysis

|  |  |  |
| --- | --- | --- |
| Delay line | Output | Comment |
| [0 0 0] | 0 | No changes are observed since all numbers are same – put 0 to as an output. |
| [0 1 0] | 0 | Now a new number is received and the oldest is discarded. The new number is higher than the rest, however, that only indicates that the input sequence is increasing and is not enough to say that a peak is detected – put 0 to as an output. |
| [0 1 0] | 1 | Here it can be noticed that the second number has the highest value. Having this in mind it can be easily concluded that a peak is detected, therefore highest value in the delay line is used to represent the amplitude of the peak, and therefore stored in the detection array. |
| [1 0 1] | 0 | No peak is detected here, at least not positive |
| [0 1 2] | 0 | Here the output is 0 again, because the samples indicate a slope going up |
| [1 2 2] | 0 | Here a flat top is detected, but because the future input cannot be predicted, a 0 is assigned to the output |
| [2 2 2] | 0 | No change here. |
| [2 2 1] | 2 | The input value is lower that the older values in the delay line, thus a peak is detected. |
| [2 1 0] | 0 | A downward slope. |
| [1 0 1] | 0 | Not a peak. |
| [0 1 0] | 1 | Once again the middle value is higher than the third (newest), therefore a peak is detected |
| [1 0 0] | 0 | Not a peak. |

The result from the table can be seen as graph on the next page.

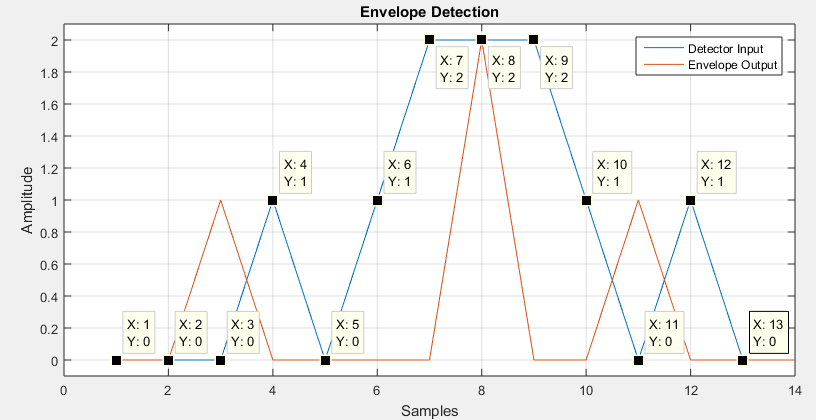


Figure detecting the peaks (orange) of the input sequence (blue)

So far so good, however, this output cannot be used as envelope of the input signal, therefore more modifications are done, which result in the following output (see figure 9)

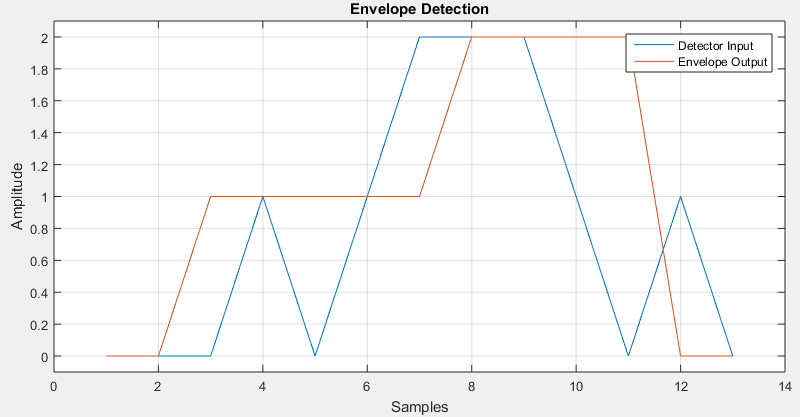


Figure Input sequence compared to the output of the modified routine

The difference between the modified version and the original is that the modified will keep the detected value until a new peak is detected, whereas the original will return 0s unless a peak is detected. A better example of the procedure is illustrated on the next image.

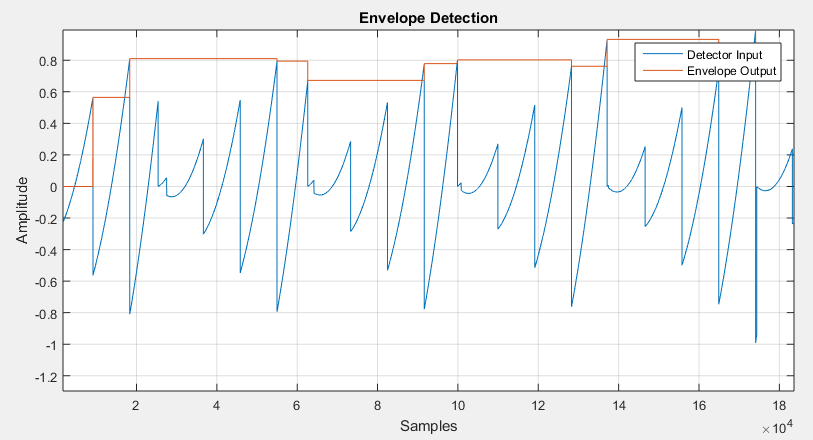


Figure Envelope detection of actual signal

On figure 10 an amplitude modulated saw tooth wave (blue) is applied to the algorithm and the output (orange) can be seen jumping from peak to peak. There is, however, one think that should be mentioned – the envelope detector does not ‘see’ all peak in the signal – it is tuned to avoid smaller peaks. This threshold is a controllable parameter. Finally this method can be used to detect the highest (or lowest) value the input have ever had.

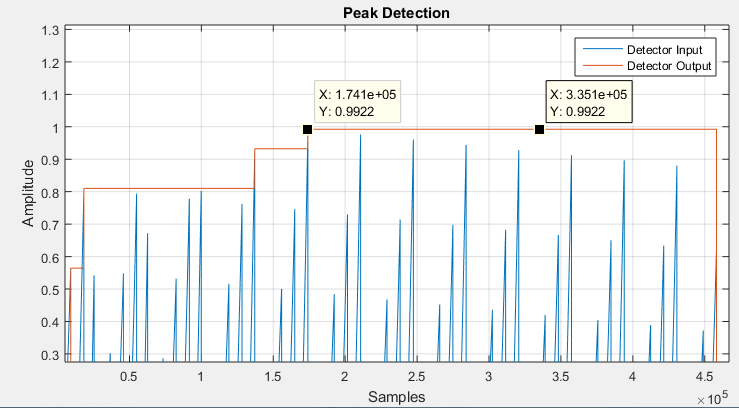


Figure Absolute maximal value detection

## The code

As mentioned the code for this procedure is made of if statements and a 3tap delay line. The MATALB code implementing this functionality can be seen below.

for l=1:N-2

del=x(l:l+2);

if del(2)>del(3)

if del(2)==del(1) || del(2)>del(1)

valn=del(2);

end

end

if valn>=valo

valo=valn;

end

env(l)=valo

end

Explanation: the loop runs reach time a sample is obtained, in this case for each element of the input array. Then the three most recent elements are being put in the delay line

del=x(l:l+2);

Next the delay line is test for the presence of peaks.

if del(2)>del(3)

if del(2)==del(1) || del(2)>del(1)

valn=del(2);

end

end

Have in mind that if a peak has a flat top (the highest value repeats multiple time one after another), the peak will be detected at its last highest value. Finally there is the threshold control, which passes only peaks with certain value

if valn>=valo\*pam

valo=valn;

end

If the new value equal or higher to the old one, only the output of the envelope will change. The parameter ‘pam’ is used only is one wants to detect peaks with lower amplitude than the already detected one. Therefore ‘pam’ should have value from 0 to 1. In that way if ‘pam’ is 0.8 then all peaks with amplitude of 80% of the current one can pass.

With further modifications the routine can detect amplitude bottoms or all-time lowest values.

1. I mentioned that this model is supposed to be used in continuous digital domain, where in theory no clocks exist, however, I would derive the sampling frequency clock for the system from the LSB of the input, this however, is another topic. [↑](#footnote-ref-1)